Programming Project – Due April 16

- 50 point project.
- Implement, in MrEd, a full Lindenmayer System parser and drawing tool based on turtle graphics.
- The turtle, in turtle graphics, is an on-screen cursor (or on-floor robot) which can be given drawing (or movement) instructions such as move forward by a specified distance or turn right or left by a specified angle.
- A state of the turtle is defined as a quadruple \((x, y, \theta, c)\).
- The Cartesian coordinates \((x, y)\) represent the turtle's position.
- The angle \(\theta\), called the heading, is interpreted as the direction in which the turtle is facing.
- The color \(c\) is interpreted as the color pen that the turtle currently has pressed to the floor so that any movement of the turtle will create a line of that color.
L-System Syntax

(L-system
  axiom; a list of atoms
  rules; a list of lists of atoms
  initial Angle; real number \([0,360]\)
  turnAngle; real number \([0, 180]\)
  shrinkage; real number \([0.01, 100.0]\)
  generations; non-negative integer.
)

L-System – Graphical Interpretation of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| f      | Draw a straight line in the direction of the current heading. 
(x, y, \(\theta\), c) \(\rightarrow\) (\(x'\), \(y'\), \(\theta\), c), where \(x' = x + d \cos(\theta)\) and \(y' = y + d \sin(\theta)\). |
| h      | Same as f. |
| g      | Same as f except no line is drawn. |
| !      | Draw line with length a function of the symbol's age. The ! symbol cannot be a variable. |
| +      | Turn clockwise by angle \(\delta\). (x, y, \(\theta\), c) \(\rightarrow\) (x, y, \(\theta + \delta\), c). |
| -      | Turn counter-clockwise by angle \(\delta\). (x, y, \(\theta\), c) \(\rightarrow\) (x, y, \(\theta - \delta\), c). |
| <      | Push current state, (x, y, \(\theta\), c), onto stack. |
| >      | Pop current state, (x, y, \(\theta\), c), from stack. |
| a, b, c...e | Change the color (x, y, \(\theta\), c) \(\rightarrow\) (x, y, \(\theta\), c'). |
L-System – ! Symbol

The ! symbol cannot be used as a variable, so once it is added to a string, it remains in the string for all future generations.

Every NEW ! symbol moves the turtle the same distance as does every f, g, and h symbol (therefore, in the equation below a new ! symbol has age=0).

Every generation that a ! symbol ages, its length is divided by the value of Shrinkage.

Thus, a Shrinkage value > 1 makes older ! symbols shorter, and a Shrinkage value < 1 makes older ! symbols longer.

The color of drawn by a ! is also a function of its age: New ! symbols are drawn in light greens. Older ! symbols are drawn with darker greens and the oldest ! symbols are drawn in browns.

\[(x, y, \theta, c) \rightarrow (x', y', \theta, c)\] where

\[x' = x + \frac{d}{s^{age}} \cos \theta \quad \quad y' = y + \frac{d}{s^{age}} \sin \theta\]

\(s\) is the shrinkage value, and \(age\) is the age of the ! symbol.

Frame Size

The initial frame size should be 800 x 500 pixels.

The drawing should be fit, without distortion, to the frame size with a few pixels of blank space around the drawing.

When the user resizes the frame, the drawing must be resized.
World → Screen Coordinates

The graphical interpretations of f, g, h and ! use the equations:

\[ y' = y + d \sin \theta \quad x' = x + d \cos \theta \]
\[ x' = x + \frac{d}{s_{ww}} \cos \theta \quad y' = y + \frac{d}{s_{ww}} \sin \theta \]

1. World coordinates are floating point numbers, screen coordinates are integer, pixel values.
2. In the world coordinate system, let \( x_0 = 0, \ y_0 = 0, \ d = 1 \).
3. Walk through the command list updating \((x, y)\), and keeping track of the minimum and maximum values of both \(x\) and \(y\) throughout the path. There is no need to draw anything on the screen during this time.
4. Using the extreme values of \(x\) and \(y\) from the walk with \(d = 1\), calculate the scaling factor and the \(x\) and \(y\) offsets for mapping world coordinates to screen coordinates.

Calculating Scaling Factor and Offsets

Assume \(\text{ScreenMinX} = 0\), \(\text{ScreenMaxX} = 100\).
Assume \(\text{WorldMinX} = -3.0\), \(\text{WorldMaxX} = 7.0\).

\[
\text{scale} = \frac{\text{ScreenWidth}}{(\text{WorldMaxX} - \text{WorldMinX})} = \frac{100}{10.0} = 10.0
\]

\[
\text{OffsetX} = -(\text{WorldMinX}) \times \text{scale} = -(-3.0) \times 10.0 = 30
\]

Example:

<table>
<thead>
<tr>
<th>Location</th>
<th>World</th>
<th>Screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>-3.0</td>
<td>0</td>
</tr>
<tr>
<td>max</td>
<td>+7.0</td>
<td>100</td>
</tr>
</tbody>
</table>

\[
\text{screenX} = \text{scale}(\text{worldX}) + \text{offsetX}
\]

<table>
<thead>
<tr>
<th>Location</th>
<th>World</th>
<th>Screen</th>
<th>(\text{screenX} = \text{scale}(\text{worldX}) + \text{offsetX})</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>-3.0</td>
<td>0</td>
<td>= 10.0 (-3.0) + 30</td>
</tr>
<tr>
<td>max</td>
<td>+7.0</td>
<td>100</td>
<td>= 10.0 (+7.0) + 30</td>
</tr>
</tbody>
</table>

Use the smaller of the \(x\)-axis and \(y\)-axis scales to scale both \(x\) and \(y\) coordinate values. This means that there may be extra blank vertical or horizontal space. To get the image in the center of this space, you will need an extra offset.

Often it is desirable to leave some blank space around all sides of an image. To do this, replace \(\text{ScreenWidth}\) with \((\text{ScreenWidth} - 2 \times \text{borderPixels})\) when calculating scale.
L-System Color Specifications

- There must be at least 6 different colors from dark brown to light green for ! symbols of different ages.
- If you have defined more generation colors for ! than there are generations in a given run, then use any subset of your generation colors.
- The colors used for a, b, c, d and e can be hard coded, and may be any set of 5 distinct colors.
- When drawing a ! symbol, its age overrides any color set by a, b, c, d, or e, but does not change the color state of the turtle.
- Background color is not specified.

L-System Error Checking Tests

Assumption: None of the test cases will have multiple errors.

1) (test msg
   (L-system '(a (b) c) '((a a a)) 1 1 1 1)
   '("Error: axiom is not a list of atoms")
   )

2) (test msg
   (L-system '() '((a a a)) 1 1 1 1)
   '("Error: axiom is not a list of atoms")
   )

3) (test msg
   (L-system '(a b c) '((a a a) (b) c) 1 1 1 1)
   '("Error: rules is not a list of a list of atoms")
   )
L-System Error Checking Tests

4) (test msg
   (L-system '(a b c) '(((a a a) (b b b) (c (c c)))
     1 1 1 1)
   '("Error: rules is not a list of a list of atoms")
)

5) (test msg
   (L-system '(a b c) '(() (b b b) (c c c)) 1 1 1 1)
   '("Error: rules is not a list of a list of atoms")
)

6) (test msg
   (L-system '(a) '(((a a a)) -370 1 1 1)
   '("Error: initialAngle must be between 0 and 360.")
)

7) (test msg
   (L-system '(a) '(((a b a)) 1 200 1 1)
   '("Error: turnAngle must be between 0 and 180.")
)

8) (test msg
   (L-system '(a) '(((a b a)) 1 110 1)
   '("Error: shrinkage must be between 0.01 and 100.")
)

9) (test msg
   (L-system '(a) '(((a b a)) 1 1 0 1)
   '("Error: shrinkage must be between 0.01 and 100.")
)

10) (test msg
   (L-system '(a) '(((a b a)) 1 1 1 1.5)
   '("Error: generations must be a non-negative integer")
)
Dragon Curve (generations 1 & 2)

The Dragon Curve is good as a first test case:
1. It is monochromatic,
2. It does not use the push and pop symbols < and >.
3. It does not use the ! symbol and therefore, does not use the shrinkage parameter.
4. All angles are 90° and all lines are parallel to either the x-axis or y-axis.
5. It is easy to generate and draw by hand.

\[\text{L-system}\]
\[
' (f) \; \text{; axiom}
' ((f f - h ) (h f + h)) \; \text{; rules}
0.0 \; \text{; initial angle}
90.0 \; \text{; turn angle}
2.0 \; \text{; shrinkage}
gen) \; \text{; generations}
\]
L-System: Open Dragon

(L-system
  'f) ;axiom
  '((f f - h ) (h f + h))
  0.0 ;initial angle
  80.0 ;turn angle
  1.0 ;shrinkage
  gen ;generations
)

Sierpiński Triangle

(L-system
  'f) ;axiom
  '((f h - f - h) ;rule
    (h f + h + f)) ;rule
  90.0 ;initial angle
  60.0 ;turn angle
  1.0 ;shrinkage
  9 ;generations
)

The Sierpiński triangle is a fractal named after Waclaw Sierpiński who described it in 1915. It was originally constructed as a curve (as it is with this L-system). It can also be constructed using the "Chaos Game", or by using an Iterated Function System.
Koch Snowflake

(L-system
  '(f + + f + + f)
  '((f a f - c f + + f - a f ) (a) (c))
  0.0 ; initial angle
  60.0 ; turn angle
  1.0 ; shrinkage
  2 ; generations
)

The Koch snowflake appeared in a 1904 paper by the Swedish mathematician Helge von Koch, and is one of the earliest fractional dimension (fractal) curves to have been described.

Koch's method of construction was to start with an equilateral triangle, then
1. divide each segment into three segments of equal length.
2. draw a new equilateral triangle that has the middle segment from step 1 as its base and points outward.
3. remove the line segment that is the base of the triangle from step 2.

Koch Snowflake with Mitsubishi Diamonds

(L-system
  '(a f + + f + + f ----b h  +  +  h  +  +  h )
  '((f f - f + + f - f) (h h + h - - h + h))
  90.0 ; initial angle
  60.0 ; turn angle
  1.0 ; shrinkage
  3 ; generations
)

Note: The colors of this drawing, as with all lines drawn by f and h symbols, are not specified. Colors a and b can be any colors that are different from each other and different from the background.
Space-Filling Peano Curve (gen 1-3)

(L-system
 'f)
 '((f a f + c h + + h - a f - - f f - c h +)
 (h - a f + c h h + + h + a f - - f - c h)
 (a) (c))
0.0 ;initial angle
60.0 ;turn angle
1.0 ;shrinkage
gen ;generations
)

Space-Filling Peano Curve (gen 4-5)

Space-filling curves or Peano curves are curves, first described by Giuseppe Peano (1858 – 1932), whose ranges contain the entire 2-dimensional unit square (or the 3-dimensional unit cube). The idea of a 1-dimensional object being space filling was found to be highly counterintuitive.

In their limit, the Koch Snowflake, the Sierpiński Triangle, and Peano curves all have infinite length, are continuous, and are nowhere differentiable.
L-System: Grass

\[
\begin{align*}
\text{(L-system)}
\quad &\text{'(x): axiom} \\
\quad &\text{'((x d f -} \\
\quad &\text{  \< \< x \> + x \> +} \\
\quad &\text{  f \< + a f x > - x) } \\
\quad &\text{(f f f)} \\
\end{align*}
\]

25.0; initial Angle
25.0; turnAngle
1.0; shrinkage
6; generations

Note: There are no ! symbols used in this system. Thus, all color comes from the a and the d symbols. I used light brown and green, but they can be any colors that are different from each other and different from the background.

L-System Test Case: 45-45-90 Tree

\[
\begin{align*}
\text{(L-system)}
\quad &\text{(x y): axiom} \\
\quad &\text{(:rules} \\
\quad &\text{  (y \< - x y > \< z x y >) } \\
\quad &\text{  (z + !)} \\
\quad &\text{  (x ! \< + + ! + + ! + + ! >)} \\
\end{align*}
\]

0; initial Angle
45; turnAngle
0.7071; shrinkage = 1/\sqrt{2}
2; generations
L-System: 45-45-90 Tree

{L-system
  (x y); axiom
  (:rules
   (y < - x y > < z x y >)
   (z + !)
   (x ! < + + ! + + ! + + ! + + ! + + ! >)
  )
  0; initial Angle
  45; turnAngle
  0.7071 ; shrinkage = 1/\sqrt{2}
  3; generations
}
L-System: 45-45-90 Tree

(L-system
  (x y); axiom
  (:rules
    (y < - x y > < z x y >)
    (z + !)
    (x ! < +++ ! +++ ! +++ ! >)
  )
0; initial Angle
45; turnAngle
0.7071; shrinkage = 1/√2
5; generations
)
L-System: 30-60-90 Tree

Axiom: xy
Rules:
y → <−−xy><zxy>
z → +!
x → <+++++++!+++!>
Initial Angle: 0°
Turn Angle: 30°
Shrinkage: 0.7071
Generation: 14

L-System Output: Fern (Generation 1, 2 & 3)

Axiom: f
Rules:
f → !<+++++f><------f>-!<++++f><------f>-!<+++f><-----f>-!f
Initial Angle: 0°
Turn Angle: 8°
Shrinkage: 0.4

Note: The main stems drawn with ! symbols and need to show color aging. The tips are drawn with f and can be any color (except the background color). All of the tips must be the same color.
L-System: Fern

(L-system
  '(f); axiom
  '(((f ! < + + + + + f >
    < - - - - - - - f >
    - ! < + + + + f >
    < - - - - - - f >
    - ! < + + + f >
    < - - - - f >
    - ! f)) ;rule
  0; initial Angle
  8; turnAngle
  0.4 ;shrinkage
  5; generations
)

L-System Weed: Generations 1, 2 & 7

(L-system
  '(b f); axiom
  '(((f ! < + f > < - f >)) ;rule
  0; initial Angle
  20; turnAngle
  0.5 ;shrinkage
  gen; generations
)

4/23/2008
### Grading Rubric

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>All source code is present and submission is easy to run and test.</td>
<td>4</td>
</tr>
<tr>
<td>Code is neat and well commented.</td>
<td>4</td>
</tr>
<tr>
<td>Pass all 10 error test cases</td>
<td>3</td>
</tr>
<tr>
<td>Dragon Curve (8 cases)</td>
<td>3</td>
</tr>
<tr>
<td>Open Dragon (2 cases)</td>
<td>3</td>
</tr>
<tr>
<td>Sierpinski Triangle (1 case)</td>
<td>3</td>
</tr>
<tr>
<td>Koch Snowflake (1 case)</td>
<td>3</td>
</tr>
<tr>
<td>Koch Snowflake with Mitsubishi (1 case)</td>
<td>3</td>
</tr>
<tr>
<td>Peano Curve (5 cases)</td>
<td>3</td>
</tr>
<tr>
<td>Grass (1 case)</td>
<td>3</td>
</tr>
<tr>
<td>45-45-90 Tree (5 cases)</td>
<td>3</td>
</tr>
<tr>
<td>30-60-90 Tree (1 case)</td>
<td>3</td>
</tr>
<tr>
<td>Fern (4 cases)</td>
<td>3</td>
</tr>
<tr>
<td>Weed (3 cases)</td>
<td>3</td>
</tr>
<tr>
<td>Image is sized to fit frame</td>
<td>3</td>
</tr>
<tr>
<td>Image resizes as frame is resized</td>
<td>3</td>
</tr>
</tbody>
</table>

**Total Points:** 50

- Any test that does not complete after 10 minutes on John's PC is considered to have failed.
- Mirror images of given output receive full credit.
- If the scaling and centering are off, there is a flat -7 penalty and no additional penalty per test case.

### Extra Credit Ideas – Up to +30 points

**Language Extensions**
1. Handling of Non-deterministic Context Free Languages (CFL).
2. Handling some subset of context sensitive languages.

**Graphical Interpretation Extensions**
1. **Collision Detection** - and perhaps some syntax that identify sections which, if any part is removed, removes the whole section.
2. **3D**: Currently every ‘+’ and ‘-’ is a right or left turn about the z-axis. Add an extension that allows rotation about not only the z-axis, but also about the x-axis and y-axis. View with an isometric or a perspective projection.
3. **Noise**: Enable the user to specify a mean and standard deviation of normally distributed noise to be added to turn angles and a separate setting for noise to be added to segment lengths.
4. **Control of segment length** in a way that extends the possible graphic objects that can be represented.
5. **Thickness**: invent a creative way to vary segments thickness in a way that allows the creation of interesting new graphic objects.
6. **Zoom**: Enable the user so somehow zoom in on small parts of an image.